CS 171: Problem Set 5

Due Date: March 7th, 2024 at 8:59pm via Gradescope

1 Insecure Candidates for MACs

Two candidate constructions of MACs are given below. The schemes use a pseudrandom function function F that maps $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$.

Show that each of the following MAC schemes is insecure.

1. Scheme 1:
   (a) $\mathsf{Gen}(1^n)$: Output $k \leftarrow \{0,1\}^n$.
   (b) $\mathsf{Mac}(k,m)$: Let $m = m_0||m_1$, where $m_0, m_1 \in \{0,1\}^n$. Then Mac outputs
   $$t := F(k,m_0)||F(k,m_0 \oplus m_1)$$
   (c) $\mathsf{Verify}(k,m,t)$: Output 1 if $t = \mathsf{Mac}(k,m)$, and output 0 otherwise.
2. Scheme 2:
   (a) $\mathsf{Gen}(1^n)$: Output $k \leftarrow \{0,1\}^n$.
   (b) $\mathsf{Mac}(k,m)$: Let $m = m_0||m_1$, where $m_0, m_1 \in \{0,1\}^{n-1}$. Then Mac samples $r \leftarrow \{0,1\}^n$, and outputs
   $$t := r||\left[F(k,r) \oplus F(k,0||m_0) \oplus F(k,1||m_1)\right]$$
   (c) $\mathsf{Verify}(k,m,t)$: Let $m = m_0||m_1$, where $m_0, m_1 \in \{0,1\}^{n-1}$, and let $t = r||t'$, where $r, t' \in \{0,1\}^n$. Output 1 if
   $$t' = F(k,r) \oplus F(k,0||m_0) \oplus F(k,1||m_1)$$
   and output 0 otherwise.

Solution

Scheme 1:

1. Construction of an adversary $\mathcal{A}$:
   (a) $\mathcal{A}$ picks arbitrary strings $m_0, m_1 \in \{0,1\}^n$ such that $m_0 \neq 0^n$, and $m_0 \neq m_1$. Then $\mathcal{A}$ queries $\mathsf{Mac}(k,\cdot)$ on two messages:
   $$m := m_0||m_0$$
   $$m' := 0^n||(m_0 \oplus m_1)$$
   (b) $\mathcal{A}$ will receive:
   $$\mathsf{Mac}(k,m) = F(k,m_0)||F(k,0^n)$$
   $$\mathsf{Mac}(k,m') = F(k,0^n)||F(k,m_0 \oplus m_1)$$

(c) $\mathcal{A}$ outputs the following message-tag pair (m^*, t^*):

$$m^* = m_0||m_1$$
$$t^* = F(k, m_0)||F(k, m_0 \oplus m_1)$$

Note that t^* can be computed from the tags that $\mathcal{A}$ received previously.

2. Analysis: First, note that $\mathsf{Verify}(k, m^*, t^*) = 1$ because $\mathsf{Mac}(k, m^*) = F(k, m_0)||F(k, m_0 \oplus m_1) = t^*$. Second, (m^*, t^*) is a valid output because $m^* \neq m$, and $m^* \neq m'$, so m^* was not previously queried.

   This means that $\mathcal{A}$ will win the MAC security game for scheme 1 with probability 1, so the scheme is insecure.

Scheme 2:

1. Construction of an adversary $\mathcal{A}$:

   (a) $\mathcal{A}$ picks an arbitrary message $m = m_0||m_1$, where $m_0, m_1 \in \{0,1\}^{n-1}$, and queries $\mathsf{Mac}(k, \cdot)$ on m. $\mathcal{A}$ receives $t = r||t'$, where

   $$t' = F(k, r) \oplus F(k, 0||m_0) \oplus F(k, 1||m_1)$$

   (b) Let r_0 be the first bit of r, and let r' be the remaining bits of r (i.e. $r = r_0||r'$).

      i. If $r_0 = 0$, then $\mathcal{A}$ outputs:

      $$m^* = r'||m_1$$
      $$t^* = 0||m_0||t'$$

      ii. If $r_0 = 1$, then $\mathcal{A}$ outputs:

      $$m^* = m_0||r'$$
      $$t^* = 1||m_1||t'$$

2. We will argue that $\mathsf{Verify}(k, m^*, t^*) = 1$ with certainty. Let $m^* = m_0^*||m_1^*$, where $m_0^*, m_1^* \in \{0,1\}^{n-1}$, and let $t^* = r^*||t^{*\prime}$, where $r^*, t^{*\prime} \in \{0,1\}^n$.

   If $r_0 = 0$, then

   $$F(k, r^*) \oplus F(k, 0||m_0^*) \oplus F(k, 1||m_1^*) = F(k, 0||m_0) \oplus F(k, r) \oplus F(k, 1||m_1) = t' = t^{*\prime}$$

   so $\mathsf{Verify}(k, m^*, t^*)$ outputs 1.

   If $r_0 = 1$, then

   $$F(k, r^*) \oplus F(k, 0||m_0^*) \oplus F(k, 1||m_1^*) = F(k, 1||m_1) \oplus F(k, 0||m_0) \oplus F(k, r) = t' = t^{*\prime}$$

   so $\mathsf{Verify}(k, m^*, t^*)$ outputs 1.

3. Next, except with negligible probability, $m^* \neq m$, so m^* is a valid output. If $r' \neq m_0$ and $r' \neq m_1$, then $m^* \neq m$. This occurs with probability $1 - \mathsf{negl}(n)$.

4. In summary, this means that $\mathcal{A}$ wins the MAC security game for scheme 2 with probability $1 - \mathsf{negl}(n)$. Therefore, scheme 2 is insecure.

■

2 Encrypt-Then-Authenticate

The encrypt-then-authenticate approach constructs a CCA-secure encryption scheme using any CPA-secure encryption scheme and any *strongly* secure MAC.[1] You will show that a MAC with regular security will not suffice.

1. Describe a MAC $\mathsf{MAC}' := (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Verify}')$ that is secure but not strongly secure. In your construction, you may start with a secure MAC, $\mathsf{MAC} := (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$.
2. Prove that MAC' is secure or cite a security proof given in discussion or lecture.[2]
3. Prove that when MAC' is combined with any CPA-secure encryption scheme using encrypt-then-authenticate, it results in an encryption scheme that is not CCA-secure.

Solution

1. Construction of MAC':
   - $\mathsf{Gen}'(1^n)$: Run $\mathsf{Gen}(1^n)$.
   - $\mathsf{Mac}'(k, m)$:
     (a) Compute $t = \mathsf{Mac}(k, m)$.
     (b) Sample $b \leftarrow \{0, 1\}$.
     (c) Output $t' := t||b$.
   - $\mathsf{Verify}'(k, m, t)$: Let $t_{\mathsf{truncated}}$ be t with the final bit removed. Run $\mathsf{Verify}(k, m, t_{\mathsf{truncated}})$, and output the result.
2. It was proven in discussion 6 that MAC' is secure.
3. We will show that when MAC' is used to construct an encryption scheme via encrypt-then-authenticate, the resulting scheme is not CCA secure.
   (a) In the CCA security game, the adversary can query the decryption oracle $\mathsf{Dec}(k, \cdot)$ even after they receive their challenge ciphertext c^*, as long as they don't query on c^* itself.
   (b) In the encrypt-then-authenticate approach, each ciphertext c has the following form:

$$\begin{aligned} c' &= \mathsf{Enc}(k_E, m) \\ t &= \mathsf{Mac}(k_M, c') \\ c &= (c', t) \end{aligned}$$

   Note that if you flip the last bit of c, then the resulting ciphertext is still a valid encryption of the same message m.

[1]The encrypt-then-authenticate approach is described in Katz & Lindell, 3rd edition, construction 5.6 and also in lecture 10, slide 21.
[2]You don't need to prove that MAC' is not strongly secure.

(c) Now, let us construct an adversary $\mathcal{A}$ that breaks the CCA security of the encryption scheme.

Description of $\mathcal{A}$:

i. Don't make any phase-I queries. Just output two challenge messages, m_0 and m_1, such that $m_0 \neq m_1$.

ii. After receiving the challenge ciphertext c^*, flip the last bit of c^* to obtain c^{**}. Query the decryption oracle $\mathsf{Dec}(k_E, \cdot)$ on c^{**} to obtain a message m'.

iii. If $m' = m_0$, then output $b' = 0$. Otherwise output $b' = 1$.

(d) We claim that $\mathcal{A}$ wins the CCA security game with probability 1.

First, $\mathcal{A}$ is allowed to query the decryption oracle on c^{**} because $c^{**} \neq c^*$. Second, c^{**} and c^* encrypt the same message, so $m' = m_b$. Then given m_b, $\mathcal{A}$ can guess b correctly with certainty.

■

3 Randomized MACs

Previously we've dealt mainly with deterministic MACs, and here we will examine one reason why: we will show that any randomized MAC can be converted into a deterministic MAC using a PRF.

A **randomized MAC** is a scheme where $\mathsf{Mac}(k, m)$ is allowed to sample a uniformly random string r each time it runs.[3] A **deterministic MAC** is a scheme where $\mathsf{Mac}(k, m)$ is a deterministic function of the inputs (k, m).

Question: Given a randomized MAC $\Pi_R = (\mathsf{Gen}_R, \mathsf{Mac}_R, \mathsf{Verify}_R)$, construct a deterministic MAC $\Pi_D = (\mathsf{Gen}_D, \mathsf{Mac}_D, \mathsf{Verify}_D)$, and prove that Π_D is secure.[4]

You may assume that Π_R takes n-bit messages and l-bit random strings, and Π_D takes n-bit messages. In your construction, you may also use a PRF F that maps $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^l$.

You may find it useful to follow the template below and fill in the blanks.[5]

1. Construction of Π_D:
   (a) $\mathsf{Gen}_D(1^n)$: Sample $k = (k_1, k_2) \leftarrow \{0,1\}^n \times \{0,1\}^n$ and output it.
   (b) $\mathsf{Mac}_D(k, m)$: Compute
   $$r = F(k_1, m) \text{ and } t = \mathsf{Mac}_R(r; k_2, m)$$
   and output t.
   (c) $\mathsf{Verify}_D(k, m, t)$: Output 1 if $\mathsf{Mac}_D(k, m) = t$, and output 0 otherwise.
2. **Claim 3.1** *Π_D is a secure MAC.*

   Proof

   (a) We will define two hybrids and show that they are indistinguishable[6]:
      i. Hyb_0 is the MAC security game $\mathsf{MAC\text{-}forge}_{\mathcal{A},\Pi_D}(n)$:
         A. Sample $k \leftarrow \mathsf{Gen}_D(1^n)$.
         B. *Query Phase*: The adversary $\mathcal{A}$ gets oracle access to $\mathsf{Mac}_D(k, \cdot)$. Let $\mathcal{Q}$ be the set of all the message queries that the adversary submits to the oracle.
         C. $\mathcal{A}$ outputs (m^*, t^*). The challenger checks that $\mathsf{Verify}_D(k, m^*, t^*) = 1$ and $m^* \notin \mathcal{Q}$. The output of the game is 1 if both checks passed and 0 otherwise.
      ii. Hyb_1 is the same as Hyb_0 except that any calls to F are replaced with calls to a uniformly random function R:

[3] We sometimes sharpen our notation from $\mathsf{Mac}(k, m)$ to $\mathsf{Mac}(r; k, m)$ to make the algorithm's random input explicit.

[4] The definition of security is given in Katz & Lindell, 3rd edition, definition 4.2 and in lecture 9, slide 5.

[5] The size of each blank box below doesn't indicate how long your answer should be. The box just marks an incomplete section of the proof, and your answers will sometimes be larger than the boxes.

[6] We've given more detail for the hybrids than necessary. The extra detail is shown in gray.

A. Sample $k \leftarrow \mathsf{Gen}_D(1^n)$, and sample a function R uniformly at random from the set of functions that map $\{0,1\}^n \to \{0,1\}^l$.

B. *Query Phase*: Let $\mathsf{Mac}'(k,m)$ be the same as $\mathsf{Mac}_D(k,m)$, except any calls to F are replaced with calls to R. Next, the adversary $\mathcal{A}$ gets oracle access to $\mathsf{Mac}'(k,\cdot)$. Finally, let $\mathcal{Q}$ be the set of all the message queries that the adversary submits to the oracle.

C. $\mathcal{A}$ outputs (m^*, t^*). The challenger checks that $\mathsf{Verify}_D(k, m^*, t^*) = 1$ and $m^* \notin \mathcal{Q}$. The output of the game is 1 if both checks passed and 0 otherwise.

(b) **Claim 3.2** *For any probabilistic polynomial-time adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\left|\Pr[\mathsf{Hyb}_0 \to 1] - \Pr[\mathsf{Hyb}_1 \to 1]\right| \le \mathsf{negl}(n)$$

Proof

i. Overview: Assume toward contradiction that for some PPT adversary $\mathcal{A}$, $\left|\Pr[\mathsf{Hyb}_0 \to 1] - \Pr[\mathsf{Hyb}_1 \to 1]\right|$ is non-negligible. Then we will use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks the PRF security of F. This is a contradiction because F is a secure PRF. Therefore, our initial assumption was false, and in fact, $\left|\Pr[\mathsf{Hyb}_0 \to 1] - \Pr[\mathsf{Hyb}_1 \to 1]\right|$ is negligible for every PPT adversary $\mathcal{A}$.

ii. Construction of $\mathcal{B}$:

A. $\mathcal{B}$ will run $\mathcal{A}$ as a subroutine and simulate Hyb_0 or Hyb_1.

B. In step A of the hybrid, $\mathcal{B}$ will not sample the key k_1 – that is the PRF challenger's job. However, $\mathcal{B}$ will sample $k_2 \leftarrow \{0,1\}^n$.

C. Whenever $\mathcal{A}$ outputs a query to $\mathsf{Mac}_D(k,\cdot)$ or $\mathsf{Mac}'(k,\cdot)$, $\mathcal{B}$ will compute the response, which is either $\mathsf{Mac}_R(F(k_1,m); k_2, m)$ or $\mathsf{Mac}_R(R(m); k_2, m)$. This entails $\mathcal{B}$ querying the oracle for $F(k_1,\cdot)$ or $R(\cdot)$.

D. Finally, the output of the hybrid is a bit b, which $\mathcal{B}$ will output as well.

iii. Pseudorandom case: $\Pr[\mathcal{B}^{F(k_1,\cdot)} \to 1] = \Pr[\mathsf{Hyb}_0 \to 1]$. This is because when $\mathcal{B}$ gets query access to $F(k_1,\cdot)$, they end up simulating Hyb_0.

iv. Truly random case: $\Pr[\mathcal{B}^{R(\cdot)} \to 1] = \Pr[\mathsf{Hyb}_1 \to 1]$. This is because when $\mathcal{B}$ gets query access to $R(\cdot)$, they end up simulating Hyb_1.

v. Therefore

$$\left|\Pr[\mathcal{B}^{F(k_1,\cdot)} \to 1] - \Pr[\mathcal{B}^{R(\cdot)} \to 1]\right| = \left|\Pr[\mathsf{Hyb}_0 \to 1] - \Pr[\mathsf{Hyb}_1 \to 1]\right|$$

which is non-negligible. This implies that $\mathcal{B}$ breaks the PRF security of F. That's a contradiction because F is secure. Therefore, our initial assumption was false, and in fact, $\left|\Pr[\mathsf{Hyb}_0 \to 1] - \Pr[\mathsf{Hyb}_1 \to 1]\right| \le \mathsf{negl}(n)$.

■

(c) **Claim 3.3** *For any probabilistic polynomial-time adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr[\mathsf{Hyb}_1 \to 1] \le \mathsf{negl}(n)$$

Proof

Lemma 3.4 $\Pr[\mathsf{Hyb}_1 \to 1] \leq \Pr[\mathsf{MAC\text{-}forge}_{\mathcal{A},\Pi_R}(n) \to 1]$

Proof

Note: We will give full credit to answers that say that Hyb_1 is the same as $\mathsf{MAC\text{-}forge}_{\mathcal{A},\Pi_R}(n)$, even though that isn't exactly correct.

i. Intuition: Hyb_1 is the same as $\mathsf{MAC\text{-}forge}_{\mathcal{A},\Pi_R}(n)$ (the MAC security game for Π_R), except that if the adversary queries the Mac oracle multiple times on the same message, then they will receive the same tag every time.[7] This is because the function R is sampled uniformly at random, so the value of $R(m)$ at any given m is a uniformly random string that is independent of $R(m')$ for any $m \neq m'$.
Intuitively, Hyb_1 is a harder game for the adversary to win because they gain no new information after the first time they query a given message.

ii. Proof Overview: We will start with an adversary $\mathcal{A}$ for Hyb_1, and then use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for $\mathsf{MAC\text{-}forge}_{\mathcal{B},\Pi_R}(n)$ with the same success probability:

$$\Pr[\mathsf{Hyb}_1 \text{ with } \mathcal{A} \text{ outputs } 1] = \Pr[\mathsf{MAC\text{-}forge}_{\mathcal{B},\Pi_R}(n) \text{ with } \mathcal{B} \text{ outputs } 1]$$

Therefore, the maximum success probability of any adversary in Hyb_1 is less than or equal to the maximum success probability of any adversary in $\mathsf{MAC\text{-}forge}_{\mathcal{B},\Pi_R}(n)$.

iii. Construction of $\mathcal{B}$:
   A. $\mathcal{B}$ runs $\mathcal{A}$ internally.
   B. From time-to-time, $\mathcal{A}$ will produce a query m for the $\mathsf{Mac}'(k,\cdot)$ oracle, and $\mathcal{B}$ will check whether m was previously queried.
      A. If not, $\mathcal{B}$ submits m as a query to the $\mathsf{Mac}_R(k_2,\cdot)$ oracle to obtain $t = \mathsf{Mac}_R(k_2, m)$ and forwards t to $\mathcal{A}$.
      B. If so, $\mathcal{B}$ responds with the same value of t that was given the last time m was queried.
   C. Eventually, $\mathcal{A}$ outputs a pair (m^*, t^*), which $\mathcal{B}$ outputs as well.

iv. $\mathcal{B}$ correctly simulates Hyb_1 because every time $\mathcal{A}$ queries a particular message m, it receives the same tag in response. So with non-negligible probability, $\mathcal{A}$ will output an (m^*, t^*) that wins the simulation of Hyb_1.
Next, any (m^*, t^*) that wins the simulation of Hyb_1 will also win $\mathsf{MAC\text{-}forge}_{\mathcal{B},\Pi_R}(n)$, so

$$\Pr[\mathsf{Hyb}_1 \text{ with } \mathcal{A} \text{ outputs } 1] = \Pr[\mathsf{MAC\text{-}forge}_{\mathcal{B},\Pi_R}(n) \text{ with } \mathcal{B} \text{ outputs } 1]$$

■

[7] In contrast, in $\mathsf{MAC\text{-}forge}_{\mathcal{A},\Pi_R}(n)$, the adversary may receive different tags every time it queries a particular message because Mac_R samples a fresh random string r for each query.

(d) Combining the parts above, we have that

$$\begin{aligned}\Pr[\mathsf{MAC\text{-}forge}_{\mathcal{A},\Pi_D}(n) \to 1] &= \Pr[\mathsf{Hyb}_0 \to 1] \\ &\leq \big|\Pr[\mathsf{Hyb}_0 \to 1] - \Pr[\mathsf{Hyb}_1 \to 1]\big| + \Pr[\mathsf{Hyb}_1 \to 1] \\ &\leq \mathsf{negl}_1(n) + \mathsf{negl}_2(n) = \mathsf{negl}_3(n)\end{aligned}$$

for some negligible functions $\mathsf{negl}_1, \mathsf{negl}_2, \mathsf{negl}_3$. Therefore, Π_D is secure.

■